Why Software is Different

by
Erik W. Aslaksen
Sinclair Knight Merz, St. Leonards, NSW 2065
easlaksen@skm.com.au

Software science and software engineering have been the originators of much of what we call systems engineering. This occurred, on the one hand, because software programs soon became so complex that their development became natural applications for the systems approach and, on the other hand, because the abstract nature of software made it relatively easy to apply the abstraction that is central to the system approach. However, if we take systems engineering to literally mean the "engineering" of systems, then we need to recognise that software development is very different from engineering and, as a consequence, carefully consider to what extent some of the software-oriented developments in systems engineering are applicable to, or appropriate for, the broader scope of engineering.

Engineering can be viewed as consisting of two distinct, but closely coupled sets of activities; the development of technology based on natural science, and the application of this technology to meet the needs of society. That is, the end objective of engineering is to create objects that provide services, and a century ago that definition would not have raised any questions. The objects were machines, boats, bridges, substances, etc., and the services they produced were clearly identifiable, even if they were embedded in a greater collection of objects providing the ultimate service. As an example, consider a newspaper. Its service has many aspects, such as bringing information to its readers, allowing businesses to reach their customer base through advertising, and so on. Producing this service requires the interplay of many elements, such as the paper, the printing machine, the journalists, and so on, but there would be no doubt about characterising the printing machine as an engineered object and a story in the paper as not. Nobody would call journalism engineering or characterise the writing of a story as text engineering.

With the advent of computers, a new dimension was introduced, in that the engineered object now consisted of two distinct parts, the hardware and the software, and they exist in a sort of symbiotic relationship, in that each part is useless by itself; only together do they provide a service. However, the nature of this relationship depends on the application; in a simple application, such as the interlocking or automation of a piece of machinery, the instructions that make up the software are identical to the hard-wired connections in a corresponding relay logic, and we would not have any hesitation in calling the development of these instructions an engineering task. But the fact that the development of the instructions or the wiring connections is carried out using a formalism, Boolean algebra, that is part of mathematics, does not make mathematics a part of engineering. We do not speak of "mathematics engineering"; engineering uses mathematics, just as we use the laws of physics, and developing the instructions using Boolean algebra is no different to using calculus to determine the strength of a shell.

If we now move to more complex applications, as the development of an accounting program or a program for calculating the strength of a shell, the relationship between hardware and software has almost disappeared. It is certainly still true that without a computer to run on the software is useless, but the developer does not have to give much, if any, consideration to the characteristics of the hardware. As long as he obeys the rules of the programming language, his program can be complied to run on any computer. And he does not have to have any domain-specific knowledge; in the case of the accounting program, accountants will specify what the program must do and the rules to be obeyed, in the case of the shell program, engineers will specify what it must do and provide all the equations etc. The situation is quite analogous to that of the journalist writing a story; he does not have to consider what paper it will be printed on or the characteristics of the printing machine, his skill lies in describing the observed facts in a manner that will meet the

readers' needs, while adhering to the rules of the language. So, why do we call the activity of the software author software *engineering*, when we do not associate any engineering with the journalist's activity?

This question is not semantic nit-picking nor an attempt at demarcation; it arises out of a concern over the current direction and focus of systems engineering, as exemplified by the recent revision of the ISO standard, ISO 15288, and the adaptation of a software modelling language, UML, to systems, as SysML. To the author, this focus seems to ignore two fundamental differences between systems engineering and software development; the level of abstraction in the subject matter itself, and the level of creativity. Engineering is about successful outcomes, and where success is measured not by one's peers, but by the stakeholders. Both the problems in executing the projects and the measures of success are often largely of a non-technical nature, and the complexity that systems engineering is called upon to handle is only partly due to the advanced technology required or the multidisciplinary nature of the project. Factors such as market forces, financial backing, politics, union pressures, environmental interest groups, personal preferences, etc. are always significant and often dominating, and these are not factors that are effectively handled by introducing any formalism or abstraction. They require extensive communication with the diverse bodies involved, and natural language is the only realistic option for this. Furthermore, the engineering of hardware is about the engineering of real, physical objects, and that activity has its own elements of abstraction and associated formalisms and languages, in the form of engineering drawings and diagrams, architectures, and ontologies (to use a very *in* word, even if a bastardisation of the philosophical meaning). Software, on the other hand, is already an abstraction; software development can be thought of as a process for expressing given dependencies between actions (or variables) in a language that can be executed by computers, and as such software development lies somewhere within the triangle spanned by mathematics, logic, and linguistics.

Not that this process does not require great skill, but it is a skill analogous to that of the journalist; of choosing the right words and constructing sentences and paragraphs that produces the correct understanding in the reader's mind of the situation being described. It also involves creativity, but it is creativity on a completely different level to that involved in engineering. Whereas creativity in engineering is concerned with creating possible solutions and then selecting the best one of these, the creativity in software development is concerned with giving the computer the most correct and efficient "explanation" of what it has to do as its part of the solution. It involves no interaction with stakeholders at all.

Both engineering and software development can be highly complex processes, and the system approach is equally applicable in both cases. In the case of engineering, the application is called systems engineering; in software development it is called something like structured programming, and while some of the basic features are the same, as is to be expected, given their common origin, there are also many features that are specific to the different natures of the two domains, as discussed above. In particular, within a given engineering project, they are located in quite different parts of the process, as illustrated in Fig. 1.

It is also worth while noting that the relative effort expended on engineering and on software development can vary greatly from project to project. On a 2 billion dollar freeway project the software development component may account for 5 million dollars and the engineering 100 million, whereas on a project to develop a new ERP module, there may be no engineering at all. In the latter case, the requirements are developed by business analysts, and the hardware platform is existing. This variability is a further indication of the relative independence of the two activities, and dispels the notion that they are two sides of the same coin and therefore need to have a common approach.

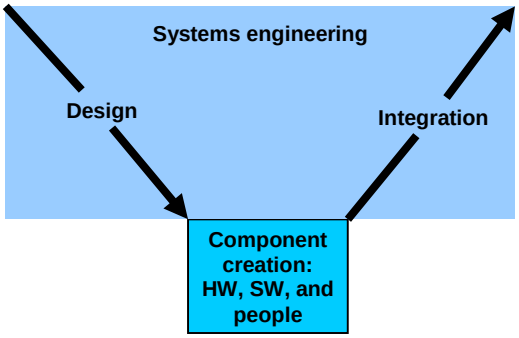

Figure 1 Systems engineering and software development are located in different areas of an engineering project.

In summary, then, the significant differences between engineering and software development in the nature of both their products and their activities give rise to doubts about the current trend in systems engineering of adopting methodologies, representations, and tools from software development without a critical examination of their applicability.